On Relationship Between a Monotone Function and the Set of its Prime Implicants in OBDD Size

AUTHOR(S):

HAYASE, Kazuyoshi

CITATION:

HAYASE, Kazuyoshi. On Relationship Between a Monotone Function and the Set of its Prime Implicants in OBDD Size. 数理解析研究所講究録 1996, 950: 1-7

ISSUE DATE:

1996-05

URL:

http://hdl.handle.net/2433/60349

数理解析研究所講究録
950巻 1996年 1-7

On Relationship Between a Monotone Function and the Set of its Prime Implicants in OBDD Size

単調関数とその主項集合を表すOBDDサイズの関係について

Kazuyoshi HAYASE (早瀬 千善)*
Department of Information Science, University of Tokyo
(東京大学大学院理学系研究科情報科学専攻)

Abstract

A state-of-the-art method for two-level logic minimization has been proposed by Coudert [3]. It uses OBDDs to represent not only Boolean functions but also the sets of their prime implicants to overcome the explosion of the number of prime implicants [4]. This method has been shown to be quite efficient in practical use but its computational complexity has been scarcely clarified. In this paper, it is shown that there exists a monotone function that has an $O(n)$ size DNF and an exponential lower bound in OBDD size, which is a solution to open questions concerned with computational complexity in [3].

1 Introduction

Computing the set of prime implicants of a Boolean function lies at the base of two-level logic minimization, which has plenty of applications in computer science. The well-known Quine-McCluskey method is the pioneer of this problem and becomes a basis of many previously known minimizers. Unfortunately, it is well-known that there are many Boolean functions in practical use which have intractably large sets of prime implicants and Quine-McCluskey based two-level logic minimizers fail to minimize them because these methods compute the sets of prime implicants explicitly.

OBDDs (Ordered Binary Decision Diagrams) [2] have been proved to be very useful in many fields such as VLSI CAD, AI and combinatorics. OBDDs have desirable properties to represent Boolean functions like: (i) OBDDs are compact for many functions in practical use, (ii) there is an efficient algorithm for Boolean operations, e.g. AND, OR, (iii) counting the number of satisfying assignments of a function can be done efficiently, and (iv) the smallest OBDD of a function is uniquely determined.

Coudert et al. have proposed a new method to overcome the explosion of prime implicants by using OBDDs [4] and their minimizer succeeds to minimize many logic circuits which could have never been treated with other methods due to the large number of prime implicants [3]. One of the key techniques of Coudert's method is that we can process the set of prime implicants implicitly by representing it with an OBDD and computational complexity has relation to not the

*E-mail: hayase@is.s.u-tokyo.ac.jp

size of that set but the size of that OBDD. As five questions have been left open in [3], however, computational complexity of this method is scarcely clarified. Followings are two of these questions: "What is the relation between the size of a sum-of-products and the size of the BDD of the function it represents?" and "What is the relation between the size of a BDD and the size of the CS (Combinational Set) of the set of its prime implicants?", where CS is a kind of OBDD which had been originally proposed in [6] and called 0-Sup-BDD (0-suppressed BDD).

In this paper, we will discuss this problem in the class of monotone (or unate) functions to make our argument not only simple but also extensible to general Boolean functions. At first, relationship in OBDD structure between a function and its prime implicants will be described. Later, we will give a solution to the two questions by showing that there exists a monotone function which has an $O(n)$ size DNF (Disjunctive Normal Form) and an exponential lower bound in OBDD size, though the second question is still left open partially.

2 Preliminaries

2.1 Monotone Boolean Function and Prime Implicant

We adopt the class of monotone Boolean functions as an analyzing tool for discussing computational complexity of the OBDD-based method for implicit computation of the set of prime implicants. Here we give some definitions and well-known properties of monotone functions.

A Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$ is assumed to have its variable set as $X = \{x_1, x_2, \ldots x_n\}$. $[f,a]$ denotes the value or subfunction of f obtained by applying a truth assignment a. For simplicity we denote truth assignments even by bit strings. The satisfying set of a function f, which is denoted by f^{-1}, is the set $\{a \in \{0,1\}^n; [f,a] = 1\}$. We often use f_j $(j = 0 \text{ or } 1)$ to denote the subfunction $[f, \{x_i := j\}]$ for readability. A Boolean function f is called *positive (negative) monotone in a variable* x_i if $(f_0)^{-1} \subseteq (f_1)^{-1}$ $((f_1)^{-1} \subseteq (f_0)^{-1})$. f is called *monotone (or unate)* if it is positive or negative monotone for all variables x_i in X. f is called positive if it is positive monotone for all variables. In place of general monotone functions, we often consider only *positive* functions without loss of generality. Next we define prime implicants of a Boolean function. A product p is a conjunction of some literals which are made from different variables each other. The satisfying set of a product p is defined as the set $p^{-1} := \{a \in \{0,1\}^n;\ [p,a] = 1\}$. A product p is an *implicant* of f if $p^{-1} \subseteq f^{-1}$. $p^{-1} \subseteq q^{-1}$. An implicant p is called a *prime implicant* of f if any other implicant q is not greater than p, that is, $p^{-1} \not\subseteq q^{-1}$. We denote the set of all the prime implicants of a function f by $PI(f)$. A prime implicant p is called *essential* if it is not covered by other prime implicants, that is, $p^{-1} \not\subseteq \bigcup_{q \in PI(f)-\{p\}} q^{-1}$. Monotone functions have the following desirable properties.

Proposition 2.1 *Any prime implicant of a monotone function f is essential. The smallest DNF (or sum-of-product form) of f is uniquely determined and furthermore it is built out of all the prime implicants of f.* □

Proposition 2.2 *f is positive (negative) monotone in x_i if and only if any prime implicant of f has no literal $\neg x_i$ (x_i).* □

We give decomposition of a function f and the set of its prime implicants $PI(f)$ in order to understand the structure of OBDDs.

The well-known *Shannon's expansion* shows the relationship of a Boolean function f with the two subfunctions concerning a variable x_i.

$$f = (\neg x_i \wedge f_0) \vee (x_i \wedge f_1) \qquad (1)$$

Some notation is necessary before stating a previously known decomposition of prime implicants. We define the product $l \wedge S$ between a literal l and a set of products S, as the set T of products which are the conjunctions l with any elements in S, that is, $T := \{p;\ p = l \wedge q,\ q \in S\}$. $PI(f)$ is known to be expanded by a variable x_i as follows [7, 1, 4]:

$$\begin{aligned} PI(f) = \ & PI(f_1 \wedge f_0) \\ \cup \ & \neg x_i \wedge \big(PI(f_0) \setminus PI(f_1 \wedge f_0)\big) \\ \cup \ & x_i \wedge \big(PI(f_1) \setminus PI(f_1 \wedge f_0)\big) \end{aligned} \qquad (2)$$

2.2 QOBDD

In this section, we briefly describe OBDDs. An OBDD is a labelled directed acyclic graph with a root [2] to represent a Boolean function. Each non-terminal node v has two outgoing edges and is labelled by a Boolean variable $label(v) \in X$. There is a total order π on the variable set X called *variable ordering*. We assume that $\pi = x_1 < \ldots < x_n$ unless otherwise specified. Each directed path follows π in the sense that $label(u) < label(v)$ if it contains an edge u to v.

By sharing isomorphic subgraphs, OBDD is known to be determined uniquely for a Boolean function and a variable ordering [2]. QOBDD (Quasi-reduced OBDD, figure 1) is obtained by applying the following rule maximally with the restriction that any directed path from the root to a terminal node contains just $n+1$ nodes. "If the two subgraphs rooted by two nodes u and v are isomorphic, redirect all incoming edges of u to v and delete u." To analyze the size of QOBDD, we consider the width of each level. The width

$$\mathtt{f} = (\neg \mathtt{x_1} \wedge \neg \mathtt{x_3}) \vee (\mathtt{x_2} \wedge \mathtt{x_3}) \vee (\mathtt{x_1} \wedge \neg \mathtt{x_2})$$

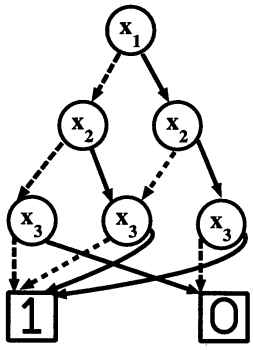

Figure 1: An Example of QOBDD

$W(D)_l$ of QOBDD D in level l is the number of the nodes which are labelled by the l-th variable of the variable ordering.

Proposition 2.3 *The width $W(D)_l$ of function f is the number of the subfunctions which are obtained by applying any truth assignments $a \in \{0,1\}^{l-1}$ to $x_1, x_2, \ldots, x_{l-1}$, that is,*

$$W(D)_l = \left|\{[f,a]; a \in \{0,1\}^{l-1}\}/=\right|$$

□

Corollary 2.1 *Let $\chi(S)$ be the characteristic function of a family S of subsets of X. The width $W(D)_l$ of $\chi(S)$ is at most the number of subsets $|S|+1$ for each l. Consequently, the size $|D|$ is at most $n(|S|+1)$.* □

3 QOBDD of $\chi(PI(f))$

In this section we treat the QOBDD of the characteristic function of the set of prime implicants of a positive function f. As we treat only positive functions, we can take into consideration only positive literals thanks to the proposition 2.2. Now we define the characteristic function $\chi(PI(f))$. We denote the positive product $p(a)$ which a represents. For example, $p(a) = x_1 \wedge x_3$ is represented by $a = [1010]$.

Definition 3.1 *For a positive function f, the characteristic function $\chi(PI(f))$ of $PI(f)$ is defined on X such that $\forall a \in \{0,1\}^n$, $[\chi(PI(f)), a] = 1$ if and only if $p(a) \in PI(f)$.* □

3.1 OBDD Structure of f and $\chi(PI(f))$

Owing to (2), we can construct a recursive procedure PRIME_POSITIVE to translate the QOBDD D of a positive function f into the QOBDD D_χ of $\chi(PI(f))$. This procedure calls internally another recursive procedure DIFF(v_1, v_2) which computes the Boolean operation $F(v_1) \wedge \neg F(v_2)$, where $F(v)$ denotes the subfunction represented by the node v. Conversely, we can translate D into D_χ by a similar procedure SUM_UP which uses not DIFF but OR internally.

procedure PRIME_POSITIVE(v)
begin
if v is terminal **then return** it;
p_0 := PRIME_POSITIVE ($edge(v, 0)$);
p_1 := PRIME_POSITIVE ($edge(v, 1)$);
return $Get(label(v), p_0$,DIFF$(p_1, p_0))$;
end;

Simple Boolean operations between two OBDDs D_1 and D_2 can be done in quadratic time of the size of OBDDs by using a 2-dimensional array [2] which is called *computed table* in literature. Still more, each node in the resultant OBDD can be regarded as a pair of nodes from D_1 and D_2. However in the case of PRIME_POSITIVE, the time complexity can not be bounded by polynomial even if we use the computed table although the number of recursive calls to PRIME_POSITIVE can be reduced to the size of the operand OBDD thanks to the computed table. The reason is that it calls another operation DIFF internally and DIFF may take nodes those which was not in the operand OBDD but which have been made by previously called DIFF's, as its operands. In fact, we can prove the next theorem which describes a node of D_χ in terms of those of D. It would be an indirect evidence of exponential time complexity of OBDD-based computation of $\chi(PI(f))$.

We need a kind of division of $PI(f)$ by a partial assignment a. This is the set of products represented by the subfunction $[\chi(PI(f)), a]$. Let $a \in \{0,1\}$ be a truth assignment of length l, and a_i represent the i-th bit of a. We denote the set of on-bits of a by $I(a)$. For example, if $a = [1010]$, $I(a) = \{1,3\}$. Conversely, for a positive product p, $A(p)$ means the smallest satisfying truth assignment. For example, $p = x_1 \wedge x_3$ makes $A(p) = [1010\ldots0]$. It should be noted that if f is positive and $[f, A(p)] = 1$ then p is an implicant of f. If a_i is in $I(a)$, "$a - a_i$" means another truth assignment which is different from a only at the i-th bit, otherwise it is not defined. Again if $a = [1010]$ then $a - a_1 = [0010]$ but $a - a_2$ is not defined. Let $PI_a(f)$ be the set of prime implicants which agree with a from x_1 to x_l. In other words, $PI_a(f) := \{p \in PI(f);\ p$ contains x_i if and only if $a_i = 1, (1 \leq \forall i \leq l)\}$. We define the division $PI(f)/a$ as the set of products which are made from products in $PI_a(f)$ by extracting the parts of $x_{l+1}, \ldots, x_n$. Namely,

$$PI(f)/a := \{[p, a];\ p \in PI_a(f)\}.$$

Theorem 3.1 *$PI(f)/a$ consists of prime implicants of the subfunction $[f, a]$ which are not prime implicants of any subfunction $[f, a - a_i]$ for $a_i = 1$. That is to say,*

$$PI(f)/a = PI([f, a]) \backslash \left(\cup_{i \in I(a)} PI([f, a - a_i]) \right).$$

Proof: ($\subseteq$) Let a product p be in $PI(f)/a$ and $p \wedge p(a)$ be an element of $PI_a(f)$. If p is not a prime implicant of $[f, a]$, then there

exists another implicant q of $[f,a]$ such that $p^{-1} \subset q^{-1}$. $q \wedge p(a)$ is also an implicant of f because f is positive. Otherwise, if p is a prime implicant of $[f, a-a_i]$ for some $i \in I(a)$, then $p_2 := [p \wedge p(a), \{x_i := 1\}]$ is also an implicant of f.
($\supseteq$) Let a product p be an element of "$PI([f,a] \setminus \cup_{a_i \in I(a)}([f, a-a_i])$," and $p \wedge p(a)$ be not a prime implicant of f. Then there exists a literal x_i of $p \wedge p(a)$ and another implicant q of f such that difference between p and q is only x_i. (i) if x_i is not included in q, $[q,a]$ is an implicant of $[f,a]$ greater than p. (ii) otherwise, $[q, a-a_i] = p$ is an implicant of $[f, a-a_i]$. (ii-a) if there is another implicant r of $[f, a-a_i]$ greater than p, r is also an implicant of $[f,a]$ because f is positive. (ii-b) p is prime in $[f, a-a_i]$, otherwise. □

From viewpoint of QOBDD, the subfunction $[\chi(PI(f)), a]$ represented by the node reached along a is equal to the next function:

$$\chi(PI([f,a])) \wedge \neg \left(\bigvee_{i \in I(a)} \chi(PI([f, a-a_i]))\right) \quad (3)$$

In other words, the node reached along a of length l in D_χ can be regarded as an $I(a)+1$-tuple of $\chi(PI([f,a^{'}]))$'s. Thus it would not be a wild estimation that the width $W(D_\chi)_l$ can become exponentially larger than the width $W(D)_l$ because the number of all the $\chi(PI([f,a^{'}]))$'s is equal to $W(D)_l$ by corollary 2.1.

Now we consider reverse relation. As we can see from theorem 3.1, $PI(f)/a$ is stolen some information on the subfunction $[f,a]$ by $PI(f)/(a-a_i)$'s and they are also stolen some information by $PI(f)/(a-a_i-a_j)$'s. We would have to consider all the $PI(f)/a^{'}$'s such that $a^{'}$ is smaller than a, this time.

Theorem 3.2 *The subfunction $[f,a]$ is described by the conjunction of all the products in $PI(f)/a^{'}$ for all $a^{'} \leq a$, that is,*

$$[f,a] = \bigvee_{a^{'} \leq a} \left(\bigvee_{p \in PI/a^{'}} p\right).$$

Sketch of proof: We can prove that any products in $PI(f)/a^{'}$ is also an implicant of $[f,a]$ and that for any supplemental assignment b of length $n-l$ such that $[f, a \cdot b] = 1$, there exists a product q in $PI(f)/a^{'}$ such that the first l bits of $A(q)$ is equal to $a^{'}$. □

In this case, the node reached along a partial assignment a in D can be regarded as a $2^{I(a)}+1$-tuple of $PI(f)/a^{'}$'s, which also gives us a negative intuition that $W(D)_l$ might become exponentially larger than $W(D_\chi)_l$. In fact, the next section exemplifies this intuition.

3.2 Solution to Coudert's Open Questions

It is pointed out that the size of the QOBDD D_χ of $\chi(PI(f))$ and that of the QOBDD D of f can differ exponentially for a variable ordering so far. In this section, we show existence of a monotone function where the size of D has an exponential lower bound in that of D_χ indeed. This implies an exponential lower bound of time complexity for SUM_UP. This fact also gives a solution to the two open questions of Coudert [3] presented in introduction, although the second question is still left open partially. This function has another important property that exponential relation holds even if we allow these two QOBDDs to have arbitrary different variable orderings respectively. It should be noted that size of a QOBDD is sensitive to variable ordering, and that there exists a function whose size changes exponentially due to variable ordering.

We treat a characteristic function of a family of vertex sets in a simple undirected graph

G. A vertex set S is independent if it contains no adjacent pair of vertices. The negation of the characteristic function of independent sets is: $\neg\chi(IS(G)) = \bigvee_{(x_i,x_j)\in E}(x_i \wedge x_j)$. Our investigation is a sequence of *mesh graphs* $\{M_k\}$ whose example is given in figure 3.2.[1] $\neg\chi(IS(M_k))$ can be expressed by an $O(n)$ size positive 2DNF because M_k has only $O(n)$ edges. Thus we have $O(n^2)$ size QOBDD D_χ of $\chi(IS(M_k))$ for any variable ordering by corollary 2.1. Now we show an exponential lower bound on D of the function $\neg\chi(IS(M_k))$ for arbitrary variable ordering π. By using same argument in [5], any subset A of vertices such that $|A| = n/2$ has at least $k/2$ of adjacent vertices outside of it, which we denote by C.

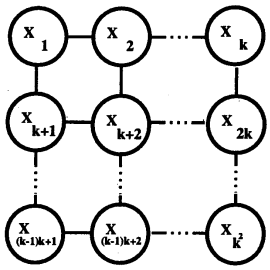

Figure 2: The mesh graph M_k

Lemma 3.1 *For any vertex subset A of M_k such that $|A| = n/2 = k^2/2$, the set C defined above satisfies $k/2 \leq |C|$.* □

Furthermore, we can find a sufficiently large subset B of A called *path collection* which has the following properties: (i) any vertex u in B has an adjacent vertex in C. (ii) any pair of two different vertices u and v in B, they do not share any adjacent vertex in C. (iii) the degree of the subgraph $M_k(B)$ (subgraph induced by B) is at most 2. (iv) the subgraph $M_k(B)$ has no cycle.

[1]Though we only show the case of the number of vertices (or variables) n is equal to k^2 for some positive integer k, this argument can be expanded to arbitrary positive integer n by ignoring some variables.

Lemma 3.2 *There exists a path collection B which contains at least $3k/640$ vertices.*

Sketch of proof: We construct B in three steps. **(Step I):** A greedy algorithm can find a subset B' of A which satisfies the properties of (i) and (ii). In this step, we consume at most 40 vertices of C per vertex of B'. **(Step II):** We remove at most half of B' to find B'' satisfying (iii). **(Step III):** We remove at most quarter of B'' to find B. It should be noted that our construction uses property of M_k like that the degree is four and that a cycle has at least four vertices. □

The positive 2DNF also confirms the fact that any two different truth assignments a_1 and a_2 to A such that (i) $v := 0$ if $v \in A - B$, and that (ii) keep independent set condition in B, induce different subfunctions. The next follows from this fact.

Lemma 3.3 *$W(D(\neg\chi(IS(M_k))))_{n/2}$ is at least the number of independent sets in the path collection B. It also equals to the arithmetic product of the numbers of independent sets in all the simple paths in B.* □

The number of independent sets in a simple path has an exponential lower bound.

Lemma 3.4 *Let $\{F_m\}$ be a Fibonacci number sequence defined by $F_1 := 2$, $F_2 := 3$ and $F_m := F_{m-1} + F_{m-2}$ for $m \geq 3$. There exist just F_m independent sets in a simple path of m vertices.* □

To sum up, we have the following.

Theorem 3.3 *There exists a monotone function f which has an $O(n)$ size DNF, and an exponential lower bound in OBDD size for arbitrary variable ordering. Furthermore, the set of its prime implicants can be expressed by polynomial size OBDD of $\chi(PI(f))$ for arbitrary variable ordering.* □

4 Conclusion

We have investigated relationship in OBDD size between a monotone function and the set of its prime implicants to give some insights into computational complexity issue of the implicit prime computation in [4].

We have shown relationship of OBDD structure between a monotone function and the characteristic function of the set of its prime implicants from which we could imagine that the time complexity of OBDD-based implicit prime computation to be exponential. Furthermore, we have found a monotone function which has a linear size DNF and can not be represented by a polynomial size OBDD whose existence had been suspected by Coudert in [3]. This example also becomes a partial solution to another open question, that is, there is a Boolean function which can not be represented by a polynomial size OBDD, while the set of prime implicants can be expressed by polynomial size OBDD for arbitrary variable ordering. Our future works are:

- Find a converse example to $\neg\chi(IS(M_k))$ to show an exponential lower bound for time complexity of PRIME_POSITIVE, or show a polynomial upper bound on OBDD size of $\chi(PI(f))$ in that of f.
- The QOBDD of maximal independent sets of M_n, or equivalently, that of $\chi(PI(\chi(IS(M_k))))$ has an exponential size QOBDD if we choose variable ordering as row-major. In other words, it has almost same size as $\chi(IS(M_k))$ in a sense. Still more it is easy to see that f, $\neg f$ and f^d (dual of f) has the same size of QOBDDs. Then it would be natural to have interests in relationship among QOBDDs of f, $\chi(PI(f))$ and $\chi(PI(\neg f))$.

Acknowledgement

The author would like to thank Associate Professor Imai and members of his laboratory for their helpful discussions and comments to this work.

References

[1] R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic Publishers, 1984.

[2] R. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, Vol. C-35, No. 8, pp. 677–691, 1986.

[3] O. Coudert. Doing two-level logic minimization 100 times faster. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pp. 112–121, 1995.

[4] O. Coudert and J. Madre. Implicit and incremental computation of primes and essential primes of Boolean functions. In *Proc. 29th ACM/IEEE DAC*, pp. 36–39, 1992.

[5] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.*, 1979.

[6] S. Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *Proc. 30th ACM/IEEE DAC*, pp. 272–277, 1993.

[7] E. Morreale. Recursive operators for prime implicant and irredundant normal form determination. *IEEE Transactions on Computers*, Vol. C-19, No. 6, pp. 504–509, 1970.